Example 48 Arithmetic Operators

```
public static void main(String[] args) {
  int max = 2147483647;
  int min = -2147483648;
  println(max+1);                          // Prints: -2147483648
  println(min-1);                          // Prints:  2147483647
  println(-min);                           // Prints: -2147483648
  print(   10/3); println(   10/(-3));     // Prints:  3 -3
  print((-10)/3); println((-10)/(-3));     // Prints: -3  3
  print(   10%3); println(   10%(-3));     // Prints:  1  1
  print((-10)%3); println((-10)%(-3));     // Prints: -1 -1
}
static void print(int i)   { System.out.print(i + " "); }
static void println(int i) { System.out.println(i + " "); }
```

Example 49 Logical Operators

Because of shortcut evaluation of `&&`, this expression from example 20 does not evaluate the array access `days[mth-1]` unless $1 \leq$ `mth` ≤ 12, so the index is never out of bounds:

```
(mth >= 1) && (mth <= 12) && (day >= 1) && (day <= days[mth-1])
```

This returns `true` if `y` is a leap year, namely, if `y` is a multiple of 4 but not of 100, or is a multiple of 400:

```
static boolean leapyear(int y)
{ return y % 4 == 0 && y % 100 != 0 || y % 400 == 0; }
```

Example 50 Bitwise Operators and Shift Operators

```
class Bitwise {
  public static void main(String[] args) throws Exception {
    int a = 0x3;                           // Bit pattern   0011
    int b = 0x5;                           // Bit pattern   0101
    println4(a);                           // Prints:       0011
    println4(b);                           // Prints:       0101
    println4(~a);                          // Prints:       1100
    println4(~b);                          // Prints:       1010
    println4(a & b);                       // Prints:       0001
    println4(a ^ b);                       // Prints:       0110
    println4(a | b);                       // Prints:       0111
  }
  static void println4(int n) {
    for (int i=3; i>=0; i--)
      System.out.print(n >> i & 1);
    System.out.println();
  }
}
```